VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREFIX RESTRICTION OF REGULATED GRAMMAR SYSTEMS

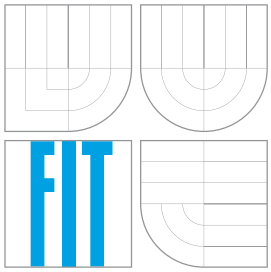
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

FILIP KONEČNÝ

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREFIXOVÉ OMEZENÍ ŘÍZENÝCH GRAMATICKÝCH SYSTÉMŮ

PREFIX RESTRICTION OF REGULATED GRAMMAR SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — FILIP KONEČNÝ
AUTHOR

VEDOUCÍ PRÁCE — prof. RNDr. ALEXANDER MEDUNA, CSc.
SUPERVISOR

BRNO 2008

Abstrakt

Tato práce studuje gramatické systémy, jejichž komponenty používají pravidla, která mají na levé straně ne jeden neterminál, ale řetězec neterminálů. Práce u těchto gramatických systémů zavádí tři omezení derivace. První vyžaduje, aby k derivaci v každé větné formě došlo v rámci prvních l symbolů v prvním spojitém bloku neterminálů. Druhé omezení definuje derivaci pro větné formy, které obsahují nejvýše m spojitých bloků neterminálů. Třetí omezení rozšiřuje druhé o podmínku, že každý takový blok může být nejvýše délky h. Hlavním výsledkem této práce jsou důkazy o zmenšení generativní síly gramatických systémů u dvou z těchto omezení.

Klíčová slova

řízené gramatické systémy, omezení derivace, prefixové omezení, generativní síla

Abstract

This thesis studies grammar systems whose components use sequences of productions whose left-hand sides are formed by nonterminal strings, not just single nonterminals. It introduces three restrictions on the derivations in these grammar systems. The first restriction requires that all rewritten symbols occur within the first l symbols of the first continuous block of nonterminals in the sentential form during every derivation step. The second restriction defines derivations over sentential forms containing no more than m continuous blocks of nonterminals. The third restriction extends the second in the way that each sequence of nonterminals must be of length h or less. As its main result, the thesis demonstrates that two of these restrictions decrease the generative power of grammar systems.

Keywords

regulated grammar systems, derivation restriction, prefix restriction, generative power

Citace

Filip Konečný: Prefix Restriction of Regulated Grammar Systems, diplomová práce, Brno, FIT VUT v Brně, 2008

Prefix Restriction of Regulated Grammar Systems

Declaration

I hereby declare that this thesis, apart from the help recognised, is my own work. Information taken from other sources and assistance received are duly acknowledged.

.....
Filip Konečný
May 15, 2008

Acknowlegements

This thesis is based on an upcoming paper which has been written jointly with prof. RNDr. Alexander Meduna, CSc., Mgr. Tomáš Masopust, Ph.D. and Jiří Šimáček, and from whose advices and recommendations I have benefited greatly. In particular, I thank prof. RNDr. Alexander Meduna, CSc. for his support during his supervision of this thesis.

Contents

Chapter 1

Introduction

Grammar systems which use type-0 components have the generative power of recursively enumerable languages. In this thesis, three closely related derivation restrictions of grammar systems are discussed. To explain these restrictions, let l be a constant. The first restriction requires that all rewritten symbols occur within the first l symbols of the first continuous block of nonterminals in the current sentential form during every derivation step. The second restriction defines derivations over sentential forms containing no more than m continuous blocks of nonterminals. The third restriction extends the second in the way that each sequence of nonterminals must be of length h or less.

As its main result, this thesis demonstrates that the first and the third restrictions decrease the generative power of grammar systems to the generative power of context-free grammars and state grammars, respectively.

This result concerning the derivation restrictions is of some interest when compared to analogical restrictions in terms of other grammars working in a context-sensitive way. Over its history, formal language theory has studied many restrictions placed on the way grammars derive sentential forms and on the forms of productions. In [9], Matthews studied derivations of grammars in the strictly leftmost (rightmost) way—that is, rewritten symbols are preceded (succeeded) only by terminals in the sentential form during the derivation. Later, in [10], he combined both approaches—leftmost and rightmost derivations—so that any sentential form during the derivation is of the form xWy, where x and y are terminal strings, W is a nonterminal string, and a production is applicable only to a leftmost or rightmost substring of W. In both cases, these restrictions result into decreasing the generative power of type-0 grammars to the power of context-free grammars.

Whereas Matthews studied restrictions placed on the forms of derivations, other authors studied the forms of productions. In [2], Book proved that if the left-hand side of any non-context-free production contains besides exactly one nonterminal only terminals, then the generative power of type-0 grammars decreases to the power of context-free grammars. He also proved that if the left-hand side of any non-context-free production has as its left context a terminal string and the left context is at least as long as the right context, then the generative power of type-0 grammars decreases to the power of context-free grammars, too. In [6], Ginsburg and Greibach proved that if the left-hand side of any production is a nonterminal string and the right-hand side contains at least one terminal, then the

generated language is context-free. Finally, in [1], Baker proved a stronger result. This result says that if any left-hand side of a production either has, besides terminals, only one nonterminal, or there is a terminal substring, β, on the right-hand side of the production such that the length of β is greater than the length of any terminal substring of the left-hand side of the production, then the generative power of type-0 grammars decreases to the power of context-free grammars. For more details, see page 198 in [12] and the literature cited there.

This thesis is structured as follows. In the second chapter, we begin with definitions of concepts of formal languages theory. In the third chapter, we formally introduce derivation restrictions studied in the thesis. The fourth chapter deals with notions related to grammar systems. In chapter five, we present the main results of study of the generative power of restricted grammar systems. In chapter six, the thesis concludes by a discussion of its findings.

Chapter 2

Preliminaries

In this chapter, definitions of formal concepts which are used in subsequent chapters are given.

2.1 Formal Languages Theory

This section introduces basic notions of formal language theory. Definitions and the notation are based on [11] where more information can be found.

2.1.1 Basic Definitions

As the first concept, an alphabet is defined. Informally, it is a collection of symbols.

Definition 2.1. An *alphabet* is a finite, non-empty set of elements called *symbols*.

Cardinality of an alphabet denotes the number of elements in it.

Definition 2.2. For an alphabet, Σ, $|\Sigma|$ denotes the *cardinality* of Σ. The *cardinality* of an alphabet is the number of symbols in Σ.

A sequence of symbols forms a *word*. The *empty word*, denoted by ε, is the word that contains no symbols. The inductive definition of a word follows.

Definition 2.3. Let Σ be an alphabet.

1. ε is a word over Σ
2. if x is a word over Σ and $a \in \Sigma$, then xa is a word over Σ

The length of a word is the number of all symbols in the word. Formal definition follows.

Definition 2.4. Let x be a word over an alphabet Σ. The *length* of x, $|x|$, is defined as follows.

1. if $x = \varepsilon$ then $|x| = 0$
2. if $x = a_1 \dots a_n$ for some $n \geq 1$, then $|x| = n$

The reversal of a word is a word written in the reverse order.

Definition 2.5. Let x be a word over an alphabet Σ. The *reversal* of x, x^R, is defined as follows.

1. if $x = \varepsilon$ then $x^R = \varepsilon$
2. if $x = a_1 \dots a_n$ for some $n \geq 1$, then $x^R = a_n \dots a_1$

Next, notions of prefix, suffix and subword are defined.

Definition 2.6. Let x and y be two words over an alphabet Σ. Then, x is a *prefix* of y if there exists a word, z, over Σ such that $xz = y$. Moreover, if $x \notin \{\varepsilon, y\}$, then x is a *proper prefix* of y. Analogically, x is a *suffix* of y if there exists a word, z, over Σ such that $zx = y$. If $x \notin \{\varepsilon, y\}$, then x is a *proper suffix* of y. Finally, x is a *subword* of y if there exist words, z and z', over Σ such that $zxz' = y$. If $x \notin \{\varepsilon, y\}$, then x is a *proper subword* of y.

Now, sets of all prefixes, suffixes and subwords of a given word are defined.

Definition 2.7. Let y be a word. Then,

- $pref(y) = \{x : x \text{ is a prefix of } y\}$
- $suf(y) = \{x : x \text{ is a suffix of } y\}$
- $sub(y) = \{x : x \text{ is a subword of } y\}$

Next, we define the *occur* function.

Definition 2.8. Let Σ be an alphabet. For a string $w \in \Sigma^*$ and $W \subseteq \Sigma$,

- $occur(w, W)$ denotes the number of occurrences of symbols from W in w

As a next notion, a language is defined. For this purpose, the set of all words over an alphabet has to be defined. Informally, a language is an arbitrary set of words over a given alphabet.

Definition 2.9. For an alphabet, Σ, Σ^* denotes the set of all words over Σ. Algebraically, Σ^* represents the free monoid generated by Σ under the operation of concatenation. The identity of Σ^* is denoted by ε. Set $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Algebraically, Σ^+ is thus the free semigroup generated by Σ.

Definition 2.10. Let Σ be an alphabet. Every subset $L \subseteq \Sigma^*$ is a *language* over Σ.

Next, basic operations on languages are defined, namely union, intersection, difference and complement.

Definition 2.11. Let L_1 and L_2 be languages. Then,

- $L_1 \cup L_2 = \{x : x \in L_1 \text{ or } x \in L_2\}$
- $L_1 \cap L_2 = \{x : x \in L_1 \text{ and } x \in L_2\}$
- $L_1 \setminus L_2 = \{x : x \in L_1 \text{ and } x \notin L_2\}$

Definition 2.12. Let L be a language over an alphabet Σ. Its complement, $\bar{L}$, is

- $\bar{L} = \Sigma^* \setminus L$

Now, we generalize notions of prefix, suffix and subword to languages:

Definition 2.13. Let Σ be an alphabet. For $\Lambda \subseteq \Sigma^*$,

- $pref(\Lambda) = \{w \in pref(w') : w' \in \Lambda\}$
- $suf(\Lambda) = \{w \in suf(w') : w' \in \Lambda\}$
- $sub(\Lambda) = \{w \in sub(w') : w' \in \Lambda\}$

2.1.2 Chomsky Hierarchy

This section describes some models related to the most famous classification of languages, Chomsky hierarchy. All models that generate languages of Chomsky language classes and some models that recognize languages of Chomsky language classes are defined.

Grammars of Chomsky Hierarchy

Firstly, definitions of a Chomsky grammar, a derivation relation and a language generated by a Chomsky grammar are given.

Definition 2.14. A *Chomsky grammar* is a quadruple $G = (N, T, S, P)$, where

- N is an alphabet of nonterminals,
- T is an alphabet of terminals such that $T \cap N = \emptyset$,
- $S \in N$ is the start nonterminal,
- $P \subseteq (N \cup T)^+ \times (N \cup T)^*$ is a finite set of productions.

Usually, we write $\alpha \to \beta$ instead of $(\alpha, \beta) \in P$.

Definition 2.15. Let G be a Chomsky grammar. If $\alpha \to \beta \in P$, $u = x_0\alpha x_1$, and $v = x_0\beta x_1$, where $x_0, x_1 \in (N \cup T)^*$, then

$$u \Rightarrow v \, [\alpha \to \beta]$$

in G or, simply, $u \Rightarrow v$. We say that G makes a *derivation step* from u to v.

Definition 2.16. Let G be a Chomsky grammar. The *language of* G is denoted by $\mathcal{L}(G)$ and defined as

$$\mathcal{L}(G) = \{w \in T^* : S \Rightarrow^* w\}.$$

Chomsky grammars can be classified into four basic groups by restrictions placed on their productions:

- Any Chomsky grammar is a type-0 (or phrase-structure) grammar.
- A grammar is a type-1 (or context-sensitive) grammar if all its productions are of the form $\alpha \to \beta$ where $|\alpha| \geq |\beta|$; with the exception of a rule $S \to \varepsilon$, where S does not appear on the right-hand side of any production.
- A grammar is a type-2 (or context-free) grammar if all its productions are of the form $\alpha \to \beta$ where $\alpha \in N$.
- A grammar is a type-3 (or regular) grammar if all its productions are of the form $\alpha \to \beta$ where $\alpha \in N$ and $\beta \in TN \cup T \cup \{\varepsilon\}$.

Notions of recursively enumerable, context-sensitive, context-free and regular languages are presented in the following definition:

Definition 2.17.

- A language, $\mathcal{L}$, is a *recursively enumerable language* if and only if $\mathcal{L} = \mathcal{L}(G)$, where G is a phrase structure grammar. Let **RE** denote the *family of recursively enumerable languages*.
- A language, $\mathcal{L}$, is a *context-sensitive language* if and only if $\mathcal{L} = \mathcal{L}(G)$, where G is a context-sensitive grammar. **CS** denote the *family of context-sensitive languages*.
- A language, $\mathcal{L}$, is a *context-free language* if and only if $\mathcal{L} = \mathcal{L}(G)$, where G is a context-free grammar. **CF** denote the *family of context-free languages*.
- A language, $\mathcal{L}$, is a *regular language* if and only if $\mathcal{L} = \mathcal{L}(G)$, where G is a regular grammar. **REG** denote the *family of regular languages*.

The following theorem shows inclusions among **REG**, **CF**, **CS** and **RE** language families. Proof can be found e.g. in [11].

Theorem 2.18.

$$\mathbf{REG} \subset \mathbf{CF} \subset \mathbf{CS} \subset \mathbf{RE}$$

This thesis deals with various restrictions which study derivations within continuous blocks of nonterminal symbols. For these purposes, we use a modified type of grammars, Ginsburg grammars, which have been proved to have the same generative power as phrase-structure grammars [5]. Formal definition of a Ginsburg grammar follows:

Definition 2.19. A Chomsky grammar is a *Ginsburg grammar* if each production $\alpha \to \beta$ satisfies $\alpha \in N^+$.

Automata of Chomsky Hierarchy

Now, we present some models for recognition of words of a given language.

Firstly, a finite automaton, which is a recognizer of regular languages, is defined.

Definition 2.20. A *finite automaton* is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of states,
- Σ is an alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$ is a finite set of rules,
- $F \subseteq Q$ is a set of final states.

Usually, we write $pa \to q \in \delta$ instead of $(p, a, q) \in \delta$.

Definition 2.21. Let M be a finite automaton. A *configuration* of M is any word from $Q\Sigma^*$. For any configuration qay, where $q \in Q$, $y \in \Sigma^*$ and any $qa \to p \in \delta$, M makes a *move* from configuration qay to configuration py according to $qa \to p$, written as

$$qay \Rightarrow py\,[qa \to p],$$

or, simply, $qay \Rightarrow py$.

Definition 2.22. Let M be a finite automaton. If $w \in \Sigma^*$ and $q_0 w \Rightarrow^* f$, where $f \in F$, then w is *accepted* by M, and $q_0 w \Rightarrow^* f$ is an *acceptance* of w in M. The language of M is defined as

$$\mathcal{L}(M) = \{w \in \Sigma^* : q_0 w \Rightarrow^* f \text{ is an acceptance of } w\}.$$

Secondly, a pushdown automaton, which is a recognizer of context-free languages, is defined.

Definition 2.23. A *pushdown automaton* is a septuple $M = (Q, \Sigma, \Omega, \delta, q_0, Z_0, F)$, where

- Q is a finite set of states,
- $\Sigma \subseteq \Omega$ is an alphabet,
- Ω is a pushdown alphabet,
- $\delta \subseteq \Omega \times Q \times \Sigma \cup \{\varepsilon\} \times \Omega^* \times Q$ is a finite set of rules,
- $q_0 \in Q$ is the initial state,
- $Z_0 \in \Omega$ is the initial pushdown symbol.
- $F \in Q$ is a set of final states,

Usually, we write $Zqa \to \gamma p$ instead of $(Z, q, a, \gamma, p) \in \delta$.

Definition 2.24. Let M be a pushdown automaton. A *configuration* of M is any word from $\Omega^* Q \Sigma^*$. For any configuration $xAqay$, where $x \in \Omega^*$, $y \in \Sigma^*$, $q \in Q$, and any $Aqa \to \gamma p \in \delta$, M makes a *move* from configuration $xAqay$ to configuration $x\gamma py$ according to $Aqa \to \gamma p$, written as

$$xAqay \Rightarrow x\gamma py\,[Aqa \to \gamma p],$$

or, simply, $xAqay \Rightarrow x\gamma py$.

Definition 2.25. Let M be a pushdown automaton. If $w \in \Sigma^*$ and $Z_0 q_0 w \Rightarrow^* f$, where $f \in F$, then w is *accepted* by M, and $Z_0 q_0 w \Rightarrow^* f$ is an *acceptance* of w in M. The language of M is defined as

$$\mathcal{L}(M) = \{w \in \Sigma^* : Z_0 q_0 w \Rightarrow^* f \text{ is an acceptance of } w\}.$$

Lastly, a Turing machine, which is a recognizer of recursively enumerable languages, is defined.

Definition 2.26. A *Turing machine* is n-tuple $M = (Q, \Sigma, \Gamma, R, s, F)$, where

- Q is a finite set of states,
- $\Sigma \subseteq \Gamma$ is an input alphabet,
- Γ is a tape alphabet; $\Delta \in \Gamma$,
- $R \subseteq Q \times \Gamma \times Q \times \Gamma$ is a relation of rules; $R = R_s \cup R_r \cup R_l$,
- $s \in Q$ is the start state,
- $F \subseteq Q$ is a set of final states.

Definition 2.27. Let M be a Turing machine. A *configuration* of M is any word χ satisfying

$$\chi \in \Gamma^* Q \Gamma^* \Delta^\omega$$

Next, let χ and χ' be configurations of M. M makes a *move* from χ to χ' according to r, written as

$$\chi \vdash \chi'[r]$$

if at least one of the following conditions holds:

- *Stationary move*

$$\chi = xpUy\Delta^\omega,\ \chi' = xqVy\Delta^\omega, \text{ and } r : (p, U, q, V) \in R_s$$

- *Right move*

$$\chi = xpUy\Delta^\omega,\ \chi' = xVqy\Delta^\omega, \text{ and } r : (p, U, q, V) \in R_r$$

- *Left move*

$$\chi = xXpUy\Delta^\omega,\ \chi' = xqXVy\Delta^\omega, \text{ and } r : (p, U, q, V) \in R_l$$

where $x, y \in \Gamma^*$, $U, V, X \in \Gamma$, $p, q \in Q$.

Definition 2.28. Let $M = (Q, \Sigma, \Gamma, R, s, F)$ be a Turing machine. M accepts $w \in \Sigma^*$ if

$$sw\Delta^\omega \vdash^* ufv\Delta^\omega$$

in M, where $f \in F$ and $u, v \in \Gamma^*$.

The *language accepted by* M is defined as

$$\mathcal{L}(M) = \{w \in \Sigma^* : \ M \text{ accepts w } \}.$$

2.1.3 State Languages

In this section, the concept of a state grammar is described. Informally, a state grammar is a context-free grammar enriched by states which make its derivations more controllable than in the case of traditional context-free grammars. Therefore, the generative power of state grammars is greater.

Definition 2.29. A *state grammar* (see [8]) is a quintuple $G = (V, W, T, P, S)$, where

- V is a total alphabet,
- W is a finite set of states,
- $T \subseteq V$ is an alphabet of terminals,
- $S \in (V \setminus T)$ is the start symbol, and
- $P \subseteq W \times (V \setminus T) \times W \times V^+$ is a finite set of productions.

Usually, we write $(q, A) \to (p, v) \in P$ instead of $(q, A, p, v) \in P$.

Definition 2.30. Let $G = (V, W, T, P, S)$ be a state grammar. For every $z \in V^*$, set ${}_G states(z) = \{q : (q, B) \to (p, v) \in P$, where $B \in (V \setminus T) \cap alph(z), v \in V^+, q, p \in W\}$.

Informally, given a state grammar $G = (V, W, T, P, S)$, ${}_G states(z)$ is a set of states. A state q is in that set if there is a left hand-side of a rule for the state q and some nonterminal symbol which is contained in a string z. The reason behind the definition of this set is following: if a state grammar G wants to make a derivation step by rewriting some symbol in a prefix of a sentential form, G has to be in some state which is an element of ${}_G states(z)$.

Definition 2.31. Let $G = (V, W, T, P, S)$ be a state grammar. If $(q, A) \to (p, v) \in P$, $x, y \in V^*$, ${}_G states(x) \cap \{q\} = \emptyset$, then G makes a derivation step from (q, xAy) to (p, xvy) in G, symbolically written as

$$(q, xAy) \Rightarrow (p, xvy) \ [(q, X) \to (p, v)].$$

In addition, if k is a positive integer such that $occur(xAy, V \setminus T) \leq k$, we say that $(q, xAy) \Rightarrow (p, xvy)[(q, A) \to (p, v)]$ is of index k, symbolically written as

$$(q, xAy) \ {}_k\Rightarrow \ (p, xvy) \ [(q, X) \to (p, v)].$$

If k is a positive integer such that $occur(xA, V \setminus T) \leq k$, we say that $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \rightarrow (p, v)]$ is of degree k, symbolically written as

$$(q, xAy) \ {}^{k}\!\Rightarrow (p, xvy) \ [(q, X) \rightarrow (p, v)].$$

To express that every derivation step in $v \Rightarrow^m w$ ($v \Rightarrow^+ w$ or $v \Rightarrow^* w$) is of index k, we write $v \ {}_k\!\Rightarrow^m w$ ($v \ {}_k\!\Rightarrow^+ w$ or $v \ {}_k\!\Rightarrow^* w$).
Analogically, to express that every derivation step in $v \Rightarrow^m w$ ($v \Rightarrow^+ w$ or $v \Rightarrow^* w$) is of degree k, we write $v \ {}^{k}\!\Rightarrow_m w$ ($v \ {}_k\!\Rightarrow^+ w$ or $v \ {}_k\!\Rightarrow^* w$).

Informally, each derivation step has to be *leftmost applicable* which means that if a nonterminal A is rewritten (under a state s) in a sentential form, there is no other nonterminal B on A's left which is rewritable under the state s, i.e. there is no rule with (s, B) as a left hand-side for every B on A's left. The set ${}_G states(z)$ perfectly suits for a definition of this restriction.

A derivation step must be made by rewriting a nonterminal which is the leftmost applicable but which does not have to be the very leftmost. The previous definition also defines a derivation relation of degree k. Informally, this means that a nonterminal being rewritten is at most k-th nonterminal from the left during every derivation step. We say that a derivation step is of index k if the sentential form which is to be rewritten contains at most k nonterminals. It clearly follows that if a derivation is of index k, it is also of degree k.

Next, a state language and a state grammar of index k and of degree k are defined.

Definition 2.32. Let $G = (V, W, T, P, S)$ be a state grammar. The set

$$\mathcal{L}(G) = \{w \in T^* \ : \ (q, S) \Rightarrow^* (p, w), q, p \in W\}$$

is called a *state language*. Furthermore, we define for every $k \geq 1$,

$$\mathcal{L}(G, \text{index } k) = \{w \in T^* \ : \ (q, S) \ {}_k\!\Rightarrow^* (p, w), q, p \in W\} \text{ and}$$

$$\mathcal{L}(G, \text{degree } k) = \{w \in T^* \ : \ (q, S) \ {}^{k}\!\Rightarrow^* (p, w), q, p \in W\}.$$

Definition 2.33. A state grammar G is of *index* k ($k \geq 1$) if

$$\mathcal{L}(G) = \mathcal{L}(G, \text{index } k).$$

A state grammar G is of *degree* k ($k \geq 1$) if

$$\mathcal{L}(G) = \mathcal{L}(G, \text{degree } k).$$

A state language L is said to be of *index* k if there is a state grammar G of index k such that $L = \mathcal{L}(G)$. A state language L is said to be of *degree* k if there is a state grammar G of degree k such that $L = \mathcal{L}(G)$.

Families of State Grammars

Definition 2.34. Let **SG** denote the family of state languages. For all $k \geq 1$, $\mathbf{SG}_k$ denotes the family of state languages of index k and $\mathbf{SG}^k$ denotes the family of state languages of degree k.

The generative power of any family of state languages of some degree is between families of context-free and context-sensitive languages. Moreover, the following inclusions among families of state languages has been established in [8].

Theorem 2.35.

$$\mathbf{CF} = \mathbf{SG}^1 \subset \mathbf{SG}^2 \subset \mathbf{SG}^3 \subset \cdots \subset \mathbf{SG}^\infty \subset \mathbf{CS}$$

The following inclusion is a corollary of theorem 2.35.

Corollary 2.36.

$$\mathbf{SG}_1 \subset \mathbf{SG}_2 \subset \mathbf{SG}_3 \subset \cdots \subset \mathbf{SG}_\infty \subset \mathbf{CS}$$

2.2 Selected Proof Techniques

This section contains descriptions of standard proof techniques which are relevant for the formal languages theory.

2.2.1 Mathematical Induction

Mathematical induction is one of the most common proof techniques. It is typically used to establish that a given statement is true for all non-negative integers or any infinite sequence starting with a number $n_0 \in \mathbb{N} \cup \{0\}$ (therefore called *induction on integers*). Another type of induction is *structural induction* which can be performed on any recursively defined concepts such as trees [7].

Standard Mathematical Induction

A proof by standard mathematical induction is done by proving that the first statement in the infinite sequence of statements is true, and then proving that if any one statement in that sequence is true, then so is the next one.

This can be expressed formally by the *axiom of induction*:

$$\forall \text{ predicates } P,\ (P(n_0) \wedge \forall m[P(m) \Rightarrow P(m+1)]) \Rightarrow \forall n P(n)$$

where P is the proposition in question and n_0, m, n are numbers from the considered infinite sequence $\{n_0, n_0+1, n_0+2, \dots\}$.

A proof by mathematical induction involves two steps:

- The *basis*—shows that $P(n_0)$ (the proposition is true for a particular integer n_0).
- *Induction step*—shows that if the proposition holds for m, then the same proposition also holds for $m+1$.

After performing these two steps, we can conclude that the proposition holds for the whole sequence.

Complete Mathematical Induction

Complete induction (or *strong induction*) is a generalization of standard induction. It differs in the induction step—we may assume not only that the proposition holds for m but also that it is true for all $m' \leq m$.

2.2.2 Proving Sets Equivalences

In formal language theory, one frequently needs to prove that a theorem which says that the sets constructed in two different ways are the same sets [7]. If S_1 and S_2 are two

expressions representing sets, proving that these sets are same (showing that $s \in S_1$ if and only if $s \in S_2$, formally $s \in S_1 \Leftrightarrow s \in S_2$), is equivalent to proving

- if $s \in S_1$, then $s \in S_2$ ($s \in S_1 \Rightarrow s \in S_2$) and
- if $s \in S_2$, then $s \in S_1$ ($s \in S_2 \Rightarrow s \in S_1$)

In formal language theory, elements of S_1 and S_2 are often languages. Then, S_1 and S_2 are called *families* (or *classes*) of languages.

2.2.3 Proofs of Equal Descriptional Power

It the formal languages theory, one often needs to prove that two types of models (such as grammars, automata) are equivalent in terms of their descriptional power.

In terms of previous section, it is possible to perceive these models as expressions representing sets (sets of strings, languages). To prove that the set of languages described by the first type of model is the same as the set of languages described by second type of model (that families of languages of these two models are equivalent), a constructive proof in combination with mathematical induction may be used. This may be done in the following manner.

Firstly, one provides general rules for the construction of an instance of second model for an arbitrary instance of the first model. To show that the constructed instance has the same descriptional power, one needs to prove two things:

- that any derivation in the instance of the second model has its counterpart in the instance of the first model
- that any derivation in the instance of the first model has its counterpart in the instance of the second model

These two proofs may be done by using mathematical induction on the length of derivation.

Secondly, one has to provide a description of the construction of an instance of first model for any instance of a second model. This is done analogically.

Chapter 3

Derivation Restrictions of Ginsburg Grammars

Derivation restrictions studied in this thesis are placed on components of grammar systems which themselves are Ginsburg grammars. This chapter introduces these restrictions.

3.1 Prefix Restriction

This section introduces the *prefix restriction* which enforces each derivation step to be performed within the first l symbols of the first continuous block of nonterminals in the sentential form. Definition of derivation relation ${}_l\Leftrightarrow$ which reflects the prefix restriction is given next.

Definition 3.1. Let $G = (N, T, S, P)$ be a Ginsburg grammar. Let $V = N \cup T$ be the total alphabet of G and let $l \geq 1$. If there is $\alpha \to \beta \in P$, $u = x_0\alpha x_1$, and $v = x_0\beta x_1$, where $x_0 \in T^*N^*$, $x_1 \in V^*$, and $\text{occur}(x_0\alpha, N) \leq l$, then

$$u \; {}_l\Leftrightarrow \; v \; [\alpha \to \beta]$$

in G or, simply, $u \; {}_l\Leftrightarrow \; v$. Let ${}_l\Leftrightarrow^k$ denote the k-fold product of ${}_l\Leftrightarrow$, where $k \geq 0$. Furthermore, let ${}_l\Leftrightarrow^*$ denote the transitive-reflexive closure of ${}_l\Leftrightarrow$.

3.2 Restrictions on Number of Nonterminal Blocks

This section introduces another derivation restrictions. The second restriction ensures that each sentential form contains at most m continuous blocks of nonterminals by defining the ${}_m\Leftrightarrow$ derivation relation. The third derivation restriction considered extends the second restriction by another requirement, namely that each continuous block of nonterminals must be of length h or less. Derivation relation reflecting this restriction is denoted as ${}^h_m\Leftrightarrow$.

Two sets used for the definition of restrictions are defined first.

Definition 3.2. Let $G = (N, T, S, P)$ be a Ginsburg grammar. Let $m, h \geq 1$. $W(m)$ denotes the set of all strings $x \in V^*$ satisfying 1 given next. $W(m, h)$ denotes the set of all strings $x \in V^*$ satisfying 1 and 2 given next.

1. $x \in (T^*N^*)^m T^*$;
2. $(y \in \text{sub}(x) \text{ and } |y| > h)$ implies $\text{alph}(y) \cap T \neq \emptyset$.

Informally, the $W(m)$ set contains all potential sentential forms that satisfy the second restriction. The $W(m, h)$ set contains all potential sentential forms that satisfy the third restriction. The first condition ensures that a sentential form has at most m continuous blocks of nonterminals. The second condition limits the length of each nonterminal block by saying that when we take any substring of a sentential form whose length is greater than h, then this substring contains some terminal symbol.

Now, both restrictions limiting number of nonterminal blocks are defined.

Definition 3.3. Let $G = (N, T, S, P)$ be a Ginsburg grammar. Let $u, v \in V^*$ and $u \Rightarrow v$.

$$u \; {}^{h}_{m}\!\!\Rightarrow v \text{ if } u, v \in W(m, h), \text{ and}$$

$$u \; {}_{m}\!\!\Rightarrow v \text{ if } u, v \in W(m).$$

Let ${}^{h}_{m}\!\!\Rightarrow^k$ and ${}_{m}\!\!\Rightarrow^k$ denote k-fold product of ${}^{h}_{m}\!\!\Rightarrow$ and ${}_{m}\!\!\Rightarrow$, respectively, where $k \geq 0$. Furthermore, let ${}^{h}_{m}\!\!\Rightarrow^*$ and ${}_{m}\!\!\Rightarrow^*$ denote the transitive-reflexive closure of ${}^{h}_{m}\!\!\Rightarrow$ and ${}_{m}\!\!\Rightarrow$, respectively.

Finally, an extension of the third restriction is defined. The extension consists in a requirement that each derivation is leftmost with respect to the set of productions currently used.

Definition 3.4. Let $G = (N, T, S, P)$ be a Ginsburg grammar. Let $x_0, x_1 \in V^*$, and $\alpha \to \beta \in P$. Then, $x_0 \alpha x_1 \; {}^{h}_{m}\!\!\Rightarrow x_0 \beta x_1$ $[\alpha \to \beta]$ is a *left-most* with respect to the set of productions P' if for each $\alpha' \to \beta' \in P'$, $x_0 \alpha x_1 = x'_0 \alpha' x'_1$ implies $|x_0| \leq |x'_0|$, where $x'_0, x'_1 \in V^*$.

3.3 Further Definitions

This section contains further definitions that are used in proofs in following chapters.

Definition 3.5. Let P be a set of productions of a grammar $G = (N, T, S, P)$. We define $N_{\text{left}}(P) = \{\alpha \in N^+ \, : \, \alpha \to \beta \in P, \beta \in (N \cup T)^*\}$.

Informally, $N_{\text{left}}(P)$ is a set of all left-hand sides of rules in P.

Chapter 4

Grammar Systems

Informally, a *grammar system* is a set of grammars which work together in order to generate one language. Two types of grammar systems are distinguished:

- *cooperating distributed (CD)* grammar systems
- *parallel communicating (PC)* grammar systems

The main difference between these two classes is that component grammars of a CD grammar system have a common sentential form, while in the case of PC grammar systems each component has its own sentential form. This thesis deals with CD grammar systems only and only this type is considered hereafter. Therefore, each use of the term *grammar system* will refer to CD grammar system hereafter. Definitions used in this chapter are based on those in [3] and [13].

4.1 Grammar Systems

Definition 4.1. A *grammar system* is a $(n+3)$-tuple $\Gamma = (N, T, S, P_1, \ldots, P_n)$, where

- N is an alphabet of nonterminal symbols,
- T is an alphabet of terminal symbols such that $N \cap T = \emptyset$; $V = N \cup T$
- $S \in N$ is the start nonterminal symbol,
- $G_i = (N, T, S, P_i), 1 \leq i \leq n$ is a Ginsburg grammar (called a *component*).

The following definition introduces derivation relations of grammar systems with respect to derivation restrictions which have been defined in the previous chapter.

Definition 4.2. Let $u, v \in V^*$, $k \geq 0$. Then, we write

$$u \; {}_l\diamond\!\Rightarrow^k_{P_i} \; v, \; u \; {}^h_m\circ\!\Rightarrow^k_{P_i} \; v, \text{ and } u \; {}_m\circ\!\Rightarrow^k_{P_i} \; v$$

to denote that

$$u \; {}_l\diamond\!\Rightarrow^k \; v, \; u \; {}^h_m\circ\!\Rightarrow^k \; v, \text{ and } u \; {}_m\circ\!\Rightarrow^k \; v,$$

respectively, was performed by P_i. Analogously, we write $u \,{}_l\diamond\!\Rightarrow^*_{P_i} v$, $u \,{}^h_m\!\circ\!\Rightarrow^*_{P_i} v$, $u \,{}_m\!\circ\!\Rightarrow^*_{P_i} v$, $u \,{}_l\diamond\!\Rightarrow^+_{P_i} v$, $u \,{}^h_m\!\circ\!\Rightarrow^+_{P_i} v$, and $u \,{}_m\!\circ\!\Rightarrow^+_{P_i} v$.

As it has been already mentioned, components of a grammar system work together on a common sentential form and generate one language. A grammar system starts a derivation with the common start nonterminal S. At each moment, only one grammar is active and is the only one that is allowed to rewrite the current sentential form at that moment. It is necessary to specify two things:

- which component can become active at a given moment
- when an activated grammar becomes inactive

Firstly, we define various modes of derivation relation of a PC grammar system with respect to restricted derivations defined in previous chapter. These modes specify the latter—when an activated grammar becomes inactive. One of possible derivation modes, k-mode, has been already defined in the definition 4.2 and refers to a situation where a component has to perform exactly k derivation steps. Following definition introduces another four derivation modes.

Definition 4.3. Let $\Gamma = (N, T, S, P_1, \ldots, P_n)$ be a grammar system. For each $i \in \{1, \ldots, n\}$ derivation modes are defined:

1. ($\leq k$-mode) $u \,{}_l\diamond\!\Rightarrow^{\leq k}_{G_i} v \iff u \,{}_l\diamond\!\Rightarrow^{k'}_{G_i} v$ for some $k' \leq k$
2. ($\geq k$-mode) $u \,{}_l\diamond\!\Rightarrow^{\geq k}_{G_i} v \iff u \,{}_l\diamond\!\Rightarrow^{k'}_{G_i} v$ for some $k' \geq k$
3. ($*$-mode) $u \,{}_l\diamond\!\Rightarrow^{*}_{G_i} v \iff u \,{}_l\diamond\!\Rightarrow^{k'}_{G_i} v$ for some $k' \geq 0$
4. (t-mode) $u \,{}_l\diamond\!\Rightarrow^{t}_{G_i} v \iff u \,{}_l\diamond\!\Rightarrow^{\geq 1}_{G_i} v$ and there is no w such that $v \,{}_l\diamond\!\Rightarrow_{G_i} w$

Analogically, we define ${}^h_m\!\circ\!\Rightarrow^{\leq k}_{G_i}$, ${}^h_m\!\circ\!\Rightarrow^{\geq k}_{G_i}$, ${}^h_m\!\circ\!\Rightarrow^{*}_{G_i}$, ${}^h_m\!\circ\!\Rightarrow^{t}_{G_i}$, ${}_m\!\circ\!\Rightarrow^{\leq k}_{G_i}$, ${}_m\!\circ\!\Rightarrow^{\geq k}_{G_i}$, ${}_m\!\circ\!\Rightarrow^{*}_{G_i}$, ${}_m\!\circ\!\Rightarrow^{t}_{G_i}$.

Informally, the $\leq k$-mode requires an active component to perform at most k derivation steps. Similarly, the $\geq k$-mode requires an active component to perform at least k derivation steps. The $*$-mode allows an active component to perform arbitrary number of derivation steps. The t-mode requires an active component to perform at least one derivation steps and also to work as long as it can—until no left-hand side of rules of the component is present in the sentential form.

Secondly, we deal with the first specification—which component can become active at a given moment. Either no limitation is given and then, we refer to such grammar systems as *unregulated grammar systems*. One can also decide to regulate an activation of components by a *control language*. Such grammar systems are called *regulated grammar systems*. These approaches are described in the two following sections.

4.2 Unregulated Grammar Systems

This section deals with *unregulated grammar systems* where no specification of which component may become active is given. Therefore, an arbitrary component can become active when an activation takes place.

The following definition of languages generated by unregulated grammar systems is generic (for an arbitrary derivation mode f and a control language L).

Definition 4.4. Let $D = \{*, t\} \cup \{\leq k, =k, \geq k \ : \ k \geq 0\}$. For a grammar system $\Gamma = (N, T, S, P_1, \ldots, P_n)$, derivation mode $f \in D$ and a control language L, we set

$$\begin{aligned}
{}_{\text{l-left}}\mathcal{L}_f^L(\Gamma) &= \{w \in T^* \ : \ S \ {}_l\Rightarrow^f_{P_{i_1}} w_1 \ {}_l\Rightarrow^f_{P_{i_2}} \cdots \ {}_l\Rightarrow^f_{P_{i_p}} w_p = w, \\
&\quad p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p\} \\
{}_{\text{nonter}}\mathcal{L}_f^L(\Gamma, m, h) &= \{w \in T^* \ : \ S \ {}^h_m\Rightarrow^f_{P_{i_1}} w_1 \ {}^h_m\Rightarrow^f_{P_{i_2}} \cdots \ {}^h_m\Rightarrow^f_{P_{i_p}} w_p = w, \\
&\quad p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p\} \\
{}_{\text{nonter}}\mathcal{L}_f^L(\Gamma, m) &= \{w \in T^* \ : \ S \ {}_m\Rightarrow^f_{P_{i_1}} w_1 \ {}_m\Rightarrow^f_{P_{i_2}} \cdots \ {}_m\Rightarrow^f_{P_{i_p}} w_p = w, \\
&\quad p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p\}.
\end{aligned}$$

Additionally, we set

$$\begin{aligned}
{}^{\text{left-most}}_{\text{nonter}}\mathcal{L}_f^L(\Gamma, m, h) &= \{w \in T^* \ : \ S \ {}^h_m\Rightarrow^f_{P_{i_1}} w_1 \ {}^h_m\Rightarrow^f_{P_{i_2}} \cdots \ {}^h_m\Rightarrow^f_{P_{i_p}} w_p = w, \\
&\quad p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p, \\
&\quad \text{and all } {}^h_m\Rightarrow_{P_{i_k}} \text{ are left-most wrt. } P_{i_k}\}
\end{aligned}$$

4.3 Regulated Grammar Systems

This section deals with *regulated grammar systems* where an order of activated components is given by a *control language*. Informally, the control language is defined over an alphabet of grammatical components of a grammar system; then, the grammar system must activate its components in such way that the sequence of components (as they are activated) must belong into that control language.

The following definition of languages generated by regulated grammar systems is generic (for an arbitrary derivation mode f and a control language L).

Definition 4.5. Let $D = \{*, t\} \cup \{\leq k, =k, \geq k \ : \ k \geq 1\}$. For a grammar system $\Gamma = (N, T, S, P_1, \ldots, P_n)$, derivation mode $f \in D$ and a control language L, we set

$$\begin{aligned}
{}_{\text{l-left}}\mathcal{L}_f^L(\Gamma) &= \{w \in T^* \ : \ S \ {}_l\Rightarrow^f_{P_{i_1}} w_1 \ {}_l\Rightarrow^f_{P_{i_2}} \cdots \ {}_l\Rightarrow^f_{P_{i_p}} w_p = w, \\
&\quad p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p, \ i_1 i_2 \ldots i_p \in L\} \\
{}_{\text{nonter}}\mathcal{L}_f^L(\Gamma, m, h) &= \{w \in T^* \ : \ S \ {}^h_m\Rightarrow^f_{P_{i_1}} w_1 \ {}^h_m\Rightarrow^f_{P_{i_2}} \cdots \ {}^h_m\Rightarrow^f_{P_{i_p}} w_p = w, \\
&\quad p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p, \ i_1 i_2 \ldots i_p \in L\} \\
{}_{\text{nonter}}\mathcal{L}_f^L(\Gamma, m) &= \{w \in T^* \ : \ S \ {}_m\Rightarrow^f_{P_{i_1}} w_1 \ {}_m\Rightarrow^f_{P_{i_2}} \cdots \ {}_m\Rightarrow^f_{P_{i_p}} w_p = w, \\
&\quad p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p, \ i_1 i_2 \ldots i_p \in L\}.
\end{aligned}$$

Additionally, we set

$$\begin{aligned}{}^{\text{left-most}}_{\text{nonter}}\mathcal{L}_f^L(\Gamma,m,h) &= \{w\in T^* : S \overset{h}{\underset{m}{\circ\Rightarrow}}{}^{f}_{P_{i_1}} w_1 \overset{h}{\underset{m}{\circ\Rightarrow}}{}^{f}_{P_{i_2}} \dots \overset{h}{\underset{m}{\circ\Rightarrow}}{}^{f}_{P_{i_p}} w_p = w,\\ &\quad p\geq 1, 1\leq i_j\leq n, 1\leq j\leq p,\ i_1i_2\dots i_p\in L,\\ &\quad \text{and all } \overset{h}{\underset{m}{\circ\Rightarrow}}{}_{P_{i_k}} \text{ are left-most wrt. } P_{i_k}\}\end{aligned}$$

It is obvious that unregulated grammar systems are a special case of regulated grammar systems where a control language contains all strings over given alphabet.

4.4 Language Families

In this section, language families of grammar systems are defined. The definition is generic (for an arbitrary derivation mode f and a family of control languages X).

Definition 4.6. Let **GS**s denote the family of all grammar systems. Let $l, m, h \geq 1$. Define the following language families:

$$\begin{aligned}{}_{\text{l-left}}GS_f^X &= \{{}_{\text{l-left}}\mathcal{L}_f^L(\Gamma) : \Gamma\in \mathbf{GS}\text{s}, L\in X\}\\ {}_{\text{nonter}}GS_f^X(m,h) &= \{{}_{\text{nonter}}\mathcal{L}_f^L(\Gamma,m,h) : \Gamma\in \mathbf{GS}\text{s}, L\in X\}\\ {}_{\text{nonter}}GS_f^X(m) &= \{{}_{\text{nonter}}\mathcal{L}_f^L(\Gamma,m) : \Gamma\in \mathbf{GS}\text{s}, L\in X\}\\ {}^{\text{left-most}}_{\text{nonter}}GS_f^X(m,h) &= \{{}^{\text{left-most}}_{\text{nonter}}\mathcal{L}_f^L(\Gamma,m,h) : \Gamma\in \mathbf{GS}\text{s}, L\in X\}\end{aligned}$$

Chapter 5

Results

This chapter presents the main results of this thesis—a study of the generative power of regulated grammar systems whose derivation relation is limited by restrictions introduced in previous chapters.

5.1 Prefix Restriction

In this section, we prove a theorem which says that the family of languages generated by grammar systems which have type-0 components, are regulated by regular languages and their derivation is restricted by the prefix restriction, is equivalent to the family of context-free languages. The proof follows the pattern described in section 2.2.2.

This section begins with with a proof of generative power for the t-mode. In its end, it discusses other derivation modes.

5.1.1 Study of the t-mode

Firstly, we deal with the ${}_{\text{l-left}}GS_t^{\mathbf{REG}} \subseteq \mathbf{CF}$ inclusion.

Proof of $\mathbf{CF} \subseteq {}_{\mathbf{l\text{-}left}}GS_t^{\mathbf{REG}}$

This inclusion is obvious. Every context-free language $\mathcal{L}$ can be modelled by a context-free grammar G. Clearly, G is a grammar system with one component. Moreover, leftmost derivation doesn't change the generative power of context-free grammars. Leftmost derivation satisfies the prefix restriction. Let the control language be $\{1\}^*$. Then, the inclusion holds.

Now, the ${}_{\text{l-left}}GS_t^{\mathbf{REG}} \subseteq \mathbf{CF}$ inclusion is proved.

Proof of CF $\supseteq$ $_{\text{l-left}}GS_t^{\mathbf{REG}}$

This section proves the second inclusion, which is formulated through the following lemma.

Lemma 5.1. *For every grammar system Γ, every finite automaton $\bar{M}$ and every $l \geq 1$, there is a pushdown automaton M, such that $\mathcal{L}(M) = {}_{l\text{-left}}\mathcal{L}_t^{\mathcal{L}(\bar{M})}(\Gamma)$.*

Proof. Let $\Gamma = (N, T, S, P_1, \ldots, P_n)$, $\bar{M} = (\bar{Q}, \bar{\Sigma}, \bar{\delta}, \bar{s_0}, \bar{F})$, and $l \geq 1$. Consider the following pushdown automaton $M = (\{s_0, f\} \cup \{[\gamma, s, \bar{s}, i] : \gamma \in N^*, |\gamma| \leq l, s \in \{q, r, e\}, \bar{s} \in \bar{Q}, i \in \{1, \ldots, n\}\}, T, T \cup N \cup \{Z\}, \delta, s_0, Z, \{f\})$, where $Z \notin T \cup N$ and δ contains rules of the following forms:

1.	$s_0 \to [S, q, \bar{s_0}, i]$	
2.	$[\gamma, q, s, i] \to (\gamma')^R[\varepsilon, r, s, i]$	if $\gamma \in N^*, \lvert\gamma\rvert \leq l$ s.t. $\gamma \; {}_l\Rightarrow^1_{P_i} \gamma'$
3.	$a[\varepsilon, r, s, i]a \to [\varepsilon, r, s, i]$	
4.	$Z[\varepsilon, r, s, i] \to f$	if $si \to s' \in \bar{\delta}$ for some $s' \in \bar{F}$
5.	$A[A_1 \ldots A_o, r, s, i] \to [A_1 \ldots A_o A, r, s, i]$	if $A \in N, o < l$
6.	$[A_1 \ldots A_l, r, s, i] \to [A_1 \ldots A_l, e, s, i]$	
7.	$a[A_1 \ldots A_o, r, s, i] \to [A_1 \ldots A_o, e, s, i]$	if $o < l, a \in T$
8.	$Z[A_1 \ldots A_o, r, s, i] \to [A_1 \ldots A_o, e, s, i]$	if $o < l$
9.	$[\gamma, e, s, i] \to [\gamma, q, s', i']$	if $sub(\gamma) \cap N_{\text{left}}(P_i) = \emptyset$, $si \to s' \in \bar{\delta}$
10.	$[\gamma, e, s, i] \to [\gamma, q, s, i]$	if $sub(\gamma) \cap N_{\text{left}}(P_i) \neq \emptyset$

We prove that $\mathcal{L}(M) = {}_{\text{l-left}}\mathcal{L}_f^{\mathcal{L}(\bar{M})}(\Gamma)$.

Proof idea. M simulates t-mode derivations of Γ regulated by $\bar{M}$ in its state which is composed of 4 elements. The first element contains first l symbols from the first continuous block of nonterminals. Third and fourth elements store a state of $\bar{M}$ and an index of an active component of Γ determined by $\bar{M}$. The second element describes a stage of a simulated derivation step; symbols in this element have the following meaning:

- q – M is ready for an application of a production of the active component,
- r – a production was applied and the first element of the state has to be updated,
- e – the first element of the state is updated and a switch to a next component of Γ can be made if the t-mode derivation of the active component is completed.

Rules of M are constructed by rule patterns 1-10. The function of these patterns is following.

Pattern 1 creates rules of M that initiate the simulation by setting the first element of state to the start nonterminal symbol, second to the stage q (ready to make a derivation step), third to the initial state of the control automaton and the fourth to an arbitrary component index.

Pattern 2 generates rules which simulate derivation steps of the grammar system. So, for arbitrary values of the third and the fourth element, the pattern creates rules which alter

first element according to production set P_i. Given a string w in the first element and a component index i in the fourth element, a rule is created for each substring of w which can be rewritten according to some production in P_i. After making a move according to one of the rules constructed in this step, third state element changes to r.

Pattern 3 generates standard popping rules of the pushdown automaton.

Pattern 4 creates rules which enable the automaton M to enter the final state provided M contains no symbols except for the special pushdown symbol Z—neither on the pushdown nor in the first state element—and provided the active component can be a suffix of a string accepted by the control automaton $\bar{M}$.

Pattern 5 generates rules which enable to load nonterminals (as many as it is possible when not exceeding the prefix limitation of l symbols) to update the monitored nonterminal sequence.

Patterns 6, 7 and 8 generate rules which enable the automaton to get to the e-stage of the simulation. Pattern 6 generates rules which allow this switch of simulation stage when the monitored nonterminal sequence is full (of length l). Pattern 7 generates rules which allow this switch of simulation stage when a terminal symbol appears on the pushdown. Pattern 8 generates rules which allow this switch of simulation stage when a special symbol Z appears on the pushdown.

Patterns 9 and 10 generate rules which allow the automaton to get to the q-stage of the simulation. Pattern 9 generates rules which allow this switch of simulation stage when an active component has completed the t-mode derivation. Rules of this type also set new active component. Pattern 10 generates rules which allow this switch of simulation stage when an active component has not completed the t-mode derivation yet. Therefore, such a component remains active.

Patterns 2 and 5 enforce the prefix restriction. The pattern 5 enables to load at most l nonterminals to the first state element and the pattern 2 creates rules which simulate derivation steps based on the content of the first element.

By this construction, simulation stages (the second state element) of the constructed automaton M can alter only in the following schematic way.

1. M makes a move according to a rule of pattern 1. M proceeds by step 2.
2. M makes a move according to a rule of pattern 2. M proceeds by step 3.
3. M makes moves according to rules of patterns 3, 4 and 5 as long as it can. If the last rule used was one constructed in 4, the simulation terminates. Otherwise, M proceeds by step 4.
4. M makes a move according to a rule of pattern 6, 7, or 8. M proceeds by step 5.
5. M makes a move according to a rule of pattern 9 or 10. M proceeds by step 2.

This simulation process can be put in more descriptive words: M monitors the first continuous block of nonterminals. If the block is longer than l symbols, it monitors only l leftmost symbols. After making a move that simulates an application of a production of a grammar

system, it has to update the monitored block. This is done by reading (popping) the terminal prefix of a string generated on the pushdown, then by adding nonterminals to the monitored prefix in the hope of extending it to the lenght l. The update ceases either when the monitored block if full (all l nonterminals have been loaded), or when a terminal occurs on pushdown (a terminal that splits the nonterminal block), or when a special pushdown symbol Z occurs. Then, when the update is finished, a possible switch in the activity of components of the grammar system is simulated. If all conditions necessary for a successful completion of the t-mode derivation are fulfilled, the switch is made. Otherwise, the activity is held by the same component.

In accordance with the section 2.2.2, it is requisite to show that the input model of the construction describe the same language as the output model. This includes showing that every derivation in an input model has its counterpart in an output model ($\supseteq$ inclusion) and vice versa ($\subseteq$ inclusion).

($\subseteq$:) The inclusion is proved by two claims. First, we prove the following claim.

Claim 5.2. *If* $Z\delta^R[\gamma, q, s, i_1]w \Rightarrow^* f$ *in* M, *then* $\gamma\delta \; {}_l\Diamond\!\!\Rightarrow^t_{P_{i_1}} w_1 \; {}_l\Diamond\!\!\Rightarrow^t_{P_{i_2}} w_2 \ldots {}_l\Diamond\!\!\Rightarrow^t_{P_{i_p}} w_p = w$, $p \geq 0$ *in* Γ *and* $i_1 \ldots i_p \in suf(\mathcal{L}(\bar{M}))$.

Proof. By induction on the number of rules constructed in 2 used in a sequence of moves.

Basis: Only one rule constructed in 2 is used. Then,

$$Z\delta^R[\gamma, q, s, i_0]w \Rightarrow Z(\gamma'\delta)^R[\varepsilon, r, s, i_0]w \Rightarrow^{|\gamma'\delta|} Z[\varepsilon, r, s, i_0] \Rightarrow f,$$

where $\gamma = \gamma_0\alpha\gamma_1$, $\gamma' = \gamma_0\beta\gamma_1$, $\alpha \to \beta \in P_{i_0}$, $\gamma \in N^+, \gamma'\delta \in T^*$. Therefore, $\gamma_0 = \gamma_1 = \varepsilon$, $\gamma'\delta = w$. Then,

$$\alpha\delta \; {}_l\Diamond\!\!\Rightarrow_{P_{i_0}} w.$$

By a rule constructed in 4 $i_0 \in suf(\mathcal{L}(\bar{M}))$ and the basis holds.

Induction hypothesis: Suppose that the claim holds for all sequences of moves containing no more than j rules constructed in 2.

Induction step: Consider a sequence of moves containing $j + 1$ rules constructed in 2:

$$\begin{array}{lll}
 & Z\delta^R[\gamma, q, s, i_0]w & \\
\Rightarrow & Z(\gamma'\delta)^R[\varepsilon, r, s, i_0]w & \text{(by a prod. constructed in 2)} \\
\Rightarrow^* & Z(\delta')^R[\varepsilon, r, s, i_0]w' & \text{(by prod. constructed in 3)} \\
\Rightarrow^* & Z(\delta'')^R[\gamma'', r, s, i_0]w' & \text{(by prod. constructed in 5)} \\
\Rightarrow & Z(\delta'')^R[\gamma'', e, s, i_0]w' & \text{(by a prod. constructed in 6, 7 or 8)} \\
\Rightarrow & Z(\delta'')^R[\gamma'', q, s', i_1]w' & \text{(by a prod. constructed in 9 or 10)} \\
\Rightarrow^* & f &
\end{array}$$

where $\gamma = \gamma_0\alpha\gamma_1$, $\gamma' = \gamma_0\beta\gamma_1$, $\alpha \to \beta \in P_{i_0}$, $\delta' \in NV^* \cup \{\varepsilon\}$, $v \in T^*$, $\gamma'\delta = v\delta'$, $vw' = w$, $\delta' = \gamma''\delta''$, either $si \to s'$ or $s = s'$, and one of the following holds:

- $|\gamma''| = l$, or
- $|\gamma''| < l$ and $\delta'' \in TV^* \cup \{\varepsilon\}$.

Then, by the rule $\alpha \to \beta$,

$$\gamma_0\alpha\gamma_1\delta \ {}_l\Leftrightarrow_{P_{i_0}} \ \gamma_0\beta\gamma_1\delta,$$

where $|\gamma_0\alpha\gamma_1| \leq l$, $\gamma_0\beta\gamma_1\delta = v\delta' = v\gamma''\delta''$ and, by the induction hypothesis,

$$v\gamma''\delta'' \ {}_l\Leftrightarrow^t_{P_{i_1}} \ vw_1 \ {}_l\Leftrightarrow^t_{P_{i_2}} \ vw_2 \dots \ {}_l\Leftrightarrow^t_{P_{i_p}} \ vw_p = vw \text{ and}$$

$$i_1 \dots i_p \in suf(\mathcal{L}(\bar{M})),$$

where $p \geq 0$.

If a rule constructed in 9 was used, $\gamma_0\alpha\gamma_1\delta \ {}_l\Leftrightarrow^t_{P_{i_0}} \ \gamma_0\beta\gamma_1\delta$ is a t-mode derivation, $i_0 i_1 i_2 \dots i_p \in suf(\mathcal{L}(\bar{M}))$ and the claim holds.

If a rule constructed in 10 was used, $i_0 = i_1$, $\gamma_0\alpha\gamma_1\delta \ {}_l\Leftrightarrow^t_{P_{i_1}} \ vw_1$, $i_1 i_2 \dots i_p \in suf(\mathcal{L}(\bar{M}))$ and the claim holds. □

Now, based on the previous claim we show that the first inclusion holds.

Claim 5.3. *If $Zs_0w \Rightarrow^* f$ in M, then $S \ {}_l\Leftrightarrow^t_{P_{i_1}} \ w_1 \ {}_l\Leftrightarrow^t_{P_{i_2}} \ w_2 \dots \ {}_l\Leftrightarrow^t_{P_{i_p}} \ w_p = w$, $p \geq 0$ in Γ and $i_1 \dots i_p \in \mathcal{L}(\bar{M})$.*

Proof. Let $Zs_0w \Rightarrow Z[S, q, \bar{s_0}, i_1]w$, by a rule constructed in 1. By the previous claim, $Z[S, q, \bar{s_0}, i_1]w \Rightarrow^* f$ implies $S \ {}_l\Leftrightarrow^t_{P_{i_1}} \ w_1 \ {}_l\Leftrightarrow^t_{P_{i_2}} \ w_2 \dots \ {}_l\Leftrightarrow^t_{P_{i_p}} \ w_p = w, p \geq 0$ in Γ and $i_1 \dots i_p \in suf(\mathcal{L}(\bar{M}))$. From the construction of 1, $i_1 \in pref(\mathcal{L}(\bar{M}))$. As $i_1 \in pref(\mathcal{L}(\bar{M}))$ and $i_1 \dots i_p \in suf(\mathcal{L}(\bar{M}))$ implies $i_1 \dots i_p \in \mathcal{L}(\bar{M})$, the claim holds. As a result, the inclusion holds. □

($\supseteq$:) The inclusion is proved by two claims. First, we prove the following claim.

Claim 5.4. *If $\tau_0 x_0 \ {}_l\Leftrightarrow^t_{P_{i_1}} \ w_1 \ {}_l\Leftrightarrow^t_{P_{i_2}} \ w_2 \dots \ {}_l\Leftrightarrow^t_{P_{i_p}} \ w_p = w$ in Γ, where $p \geq 0$, $\tau_0 \in N^+$, $x_0 \in TV^* \cup \{\varepsilon\}$, $w_i \in V^*$, $i \in \{1, \dots, p-1\}$, $w_p, w \in T^*$ and $i_1 \dots i_p \in suf(\mathcal{L}(\bar{M}))$, then $Z(\tau_0^2 x_0)^R[\tau_0^1, q, s, i_1]w \Rightarrow^* f$, for some $s \in \bar{Q}$, where $\tau_0 = \tau_0^1\tau_0^2$, $|\tau_0| \leq l$ implies $\tau_0^1 = \tau_0$, and $|\tau_0| > l$ implies $|\tau_0^1| = l$.*

Proof. By induction on the length of derivations.

Basis: Let $\tau_0 x_0 \ {}_l\Leftrightarrow_{P_{i_0}} \ \tau_0' x_0 = w$, where $\tau_0^1 = \gamma_0\alpha\gamma_1$, $\tau_0' = \gamma_0\beta\gamma_1\tau_0^2$, $\alpha \to \beta \in P_{i_0}$, $\tau_0' x_0 \in {}_{\text{l-left}}\mathcal{L}_f^{\mathcal{L}(\bar{M})}(\Gamma)$. Therefore, $\gamma_0 = \gamma_1 = \tau_0^2 = \varepsilon$. M simulates this derivation step in the following way:

$$
\begin{array}{lll}
 & Z(\tau_0^2 x_0)^R[\tau_0^1, q, s, i_0]w & \\
\Rightarrow & Z(\tau_0' x_0)^R[\varepsilon, r, s, i_0]w & \text{(by a prod. constructed in 2)} \\
\Rightarrow^{|\tau_0' x_0|} & Z[\varepsilon, r, s, i_0] & \text{(by prod. constructed in 3)} \\
\Rightarrow & f & \text{(by a prod. constructed in 4)}
\end{array}
$$

for some $s \in \bar{Q}$ (follows from construction of 4). Therefore, the basis holds.

Induction hypothesis: Suppose that the claim holds for all derivations of length j or less.

Induction step: Consider a derivation of length $j+1$:

$$\tau_0 x_0 \ {}_l\Leftrightarrow_{P_{i_0}} \tau_0' x_0 = v_1\tau_1 x_1 \ {}_l\Leftrightarrow^t_{P_{i_1}} v_1 w_1 \ {}_l\Leftrightarrow^t_{P_{i_2}} w_2 \ldots {}_l\Leftrightarrow^t_{P_{i_p}} w_p = w = v_1 w',$$

where $p \geq 0$, $v_1 \in T^*$, $\tau_0, \tau_1 \in N^+$, $\tau_0' \in V^*$, $x_0, x_1 \in TV^* \cup \{\varepsilon\}$, $w_i \in V^*, i \in \{1, \ldots, p-1\}$, $w_p, w' \in T^*$. Then, M simulates this derivation as follows:

$$
\begin{array}{lll}
 & Z(\tau_0^2 x_0)^R[\tau_0^1, q, s, i_0]w & \\
\Rightarrow & Z(\tau_0' x_0)^R[\varepsilon, r, s, i_0]w & \text{(by a prod. constructed in 2)} \\
= & Z(v_1\tau_1 x_1)^R[\varepsilon, r, s, i_0]v_1 w' & \\
\Rightarrow^{|v_1|} & Z(\tau_1 x_1)^R[\varepsilon, r, s, i_0]w' & \text{(by prod. constructed in 3)} \\
\Rightarrow^{|\tau_1^1|} & Z(\tau_1^2 x_1)^R[\tau_1^1, r, s, i_0]w' & \text{(by prod. constructed in 5)} \\
\Rightarrow & Z(\tau_1^2 x_1)^R[\tau_1^1, e, s, i_0]w' & \text{(by a prod. constructed in 6, 7, or 8)} \\
\Rightarrow & Z(\tau_1^2 x_1)^R[\tau_1^1, q, s', i_1]w' & \text{(by a prod. constructed in 9 or 10)} \\
\Rightarrow^* & f & \text{(by the induction hypothesis)}
\end{array}
$$

If $\tau_0 x_0 \ {}_l\Leftrightarrow_{P_{i_0}} \tau_0' x_0$ is a t-mode derivation, a rule of type 9 is used during the simulation. Otherwise, a rule of type 10 is used (and therefore $i_0 = i_1$). Hence, the claim holds. □

Now, based on the previous claim we show that the second inclusion holds. Validity of the lemma 5.1 is then a direct consequence.

Claim 5.5. *If* $S \ {}_l\Leftrightarrow_{P_{i_0}} u\tau_0 x_0 \ {}_l\Leftrightarrow^t_{P_{i_1}} u\varphi_1 \ldots {}_l\Leftrightarrow^t_{P_{i_p}} u\varphi_p = uw$, *where* $p \geq 0$, $u, \varphi_p, w \in T^*$, $\tau_0 \in N^+ \cup \{\varepsilon\}$, $x_0 \in TV^* \cup \{\varepsilon\}$, $\varphi_i \in V^*$, $i \in \{1, ..p-1\}$ *in* Γ *and* $i_1 \ldots i_p \in \mathcal{L}(\bar{M})$, *then* $Zs_0 w \Rightarrow^* f$ *in* M.

Proof. If $u\tau_0 x_0 \notin T^*$, M simulates this derivation in the following way:

$$
\begin{array}{lll}
 & Zs_0 uw & \\
\Rightarrow & Z[S, q, s, i_0] & \text{(by a prod. constructed in 1)} \\
\Rightarrow & Z(u\tau_0 x_0)^R[\varepsilon, r, s, i_0]uw & \text{(by a prod. constructed in 2)} \\
\Rightarrow^{|u|} & Z(\tau_0 x_0)^R[\varepsilon, r, s, i_0]w & \text{(by prod. constructed in 3)} \\
\Rightarrow^{|\tau_0^1|} & Z(\tau_0^2 x_0)^R[\tau_0^1, r, s, i_0]w & \text{(by prod. constructed in 5)} \\
\Rightarrow & Z(\tau_0^2 x_0)^R[\tau_0^1, e, s, i_0]w & \text{(by a prod. constructed in 6, 7, or 8)} \\
\Rightarrow & Z(\tau_0^2 x_0)^R[\tau_0^1, q, s', i_1]w & \text{(by a prod. constructed in 9 or 10)} \\
\Rightarrow^* & f & \text{(by the previous claim)}
\end{array}
$$

If $u\tau_0 x_0 \in T^*$, M simulates this derivation in the following way:

	Zs_0uw	
$\Rightarrow$	$Z[S, q, s, i_0]$	(by a prod. constructed in 1)
$\Rightarrow$	$Z(u\tau_0 x_0)^R[\varepsilon, r, s, i_0]uw$	(by a prod. constructed in 2)
$\Rightarrow^{\lvert u\tau_0 x_0\rvert}$	$Z[\varepsilon, r, s, i_0]w$	(by prod. constructed in 3)
$\Rightarrow$	f	(by a prod. constructed in 4)

Hence, the claim holds. As a result, the inclusion holds. □

From the previous claims, it follows that the lemma holds. □

Theorem 5.6. *Let l be a positive integer. Then,* $\mathbf{CF} = {}_{l\text{-left}}GS_t^{\mathbf{REG}}$.

Proof. One inclusion follows from Lemma 5.1, the other is clear (the argument is given in section 5.1.1). □

5.1.2 Study of Other Derivation Modes

This section concerns with a discussion of the generative power of regulated grammar systems with the prefix restriction on its derivation for the rest of possible modes of grammar system. Analogically to the discussion of t-mode, the rule patterns are given. Unlike the t-mode discussion, the full formal proof is omitted as it would be very similar to the t-mode proof. The reason for such a similarity is that only some rule patterns have to be altered. The semantics of rules is nearly the same, so the structure of the proof would also be the same. The following sections discuss rule patterns for various derivation modes briefly.

Notes on k-mode

For the purposes of k-mode, the fifth element of a pushdown state must be added. Its task is to count number of simulated derivation steps performed by an active component (i.e. number of moves according to pattern 2). For clarity, the full list of rule patterns is given.

1.	$s_0 \to [S, q, \bar{s_0}, i, 0]$	
2.	$[\gamma, q, s, i, h] \to (\gamma')^R[\varepsilon, r, s, i, h+1]$	if $\gamma \in N^*, \lvert\gamma\rvert \le l$ s.t. $\gamma\ {}_l\Leftrightarrow^1_{P_i}\ \gamma'$
3.	$a[\varepsilon, r, s, i, h]a \to [\varepsilon, r, s, i, h]$	
4.	$Z[\varepsilon, r, s, i, h] \to f$	if $si \to s' \in \bar{\delta}$ for some $s' \in \bar{F}$, $h = k$
5.	$A[A_1 \dots A_o, r, s, i, h] \to [A_1 \dots A_o A, r, s, i, h]$	if $A \in N, o < l$
6.	$[A_1 \dots A_l, r, s, i, h] \to [A_1 \dots A_l, e, s, i, h]$	
7.	$a[A_1 \dots A_o, r, s, i, h] \to [A_1 \dots A_o, e, s, i, h]$	if $o < l, a \in T$
8.	$Z[A_1 \dots A_o, r, s, i, h] \to [A_1 \dots A_o, e, s, i, h]$	if $o < l$
9.	$[\gamma, e, s, i, h] \to [\gamma, q, s', i', 0]$	if $sub(\gamma) \cap N_{\text{left}}(P_i) = \emptyset$, $si \to s' \in \bar{\delta}$, $h = k$
10.	$[\gamma, e, s, i, h] \to [\gamma, q, s, i, h]$	if $sub(\gamma) \cap N_{\text{left}}(P_i) \neq \emptyset$ and $h \neq k$

Only rules constructed in 1, 2 and 9 alter the new element. Rules of pattern 1 initiate it to zero, rules of pattern 2 increment it by one, rules of pattern 9 reset it to zero. There are two moments when it is necessary to ensure that the mode is respected. First is when passing the activity from one component to another (rules of pattern 9), second when the simulation is about to terminate (rules of pattern 4).

Notes on $\geq k$-mode

When dealing with the $\geq k$-mode, conditions on the fifth element used in rule patterns for the k-mode must be altered:

- pattern 4 – for $h \geq k$ instead of $h = k$
- pattern 9 – for $h \geq k$ instead of $h = k$
- pattern 10 – for arbitrary h instead of $h \neq k$

Notes on $\leq k$-mode

Similarly to the $\geq k$-mode, dealing with the $\leq k$-mode involves few alteration of conditions on the fifth element used in rule patterns:

- pattern 4 – for $h \leq k$ instead of $h = k$
- pattern 9 – for $h \leq k$ instead of $h = k$
- pattern 10 – for arbitrary h instead of $h \neq k$

Notes on $*$-mode

The $*$-mode, which is normally considered, does not make so much sense when considering grammar systems which are regulated. If this mode was applied to regulated grammar systems, components which were activated by a finite automaton could perform zero derivation steps and they would fulfill the definition of $*$-mode. As this would be slightly unusual way of understanding the work of an active component, this mode has not been considered. On the contrary, $^+$-mode perfectly makes sense. $^+$-mode is a special case of $\geq k$-mode (namely ≥ 1-mode) which has been already described.

Summary

We conclude this chapter by stating that when an prefix restriction is applied on regulated grammar systems, their generative power decreases to family of context-free languages for the following modes of derivation: t-mode, k-mode, $\geq k$-mode and $\leq k$-mode.

In brief, we can summarize these statements in one theorem.

Theorem 5.7. *Let l be a positive integer. Then,* $\mathbf{CF} = {}_{l\text{-}left}GS_f^{\mathbf{REG}}$ *for all* $f \in \{t\} \cup \{\leq k, =k, \geq k : k \geq 1\}$.

5.2 Restriction on Number of Nonterminal Blocks

We show that the restriction studied in this section does not decrease the generative power of regulated grammar systems and so they are still able to generate all type-0 languages. The proof shows that every type-0 language can be generated by a grammar which ensures that each sentential form contains only one continuous nonterminal block and therefore it is a special case of a grammar system with restricted derivation which proves the first inclusion. The second inclusion—which states that a language generated by any regulated grammar is a type-0 language—is obvious and is omitted in the proof. Indeed, by the Church-Turing thesis, any regulated grammar system can be simulated by a Turing machine.

Theorem 5.8. $\mathbf{RE} = {}_{nonter}GS_t^{\mathbf{REG}}(1)$.

Proof. It is well-known (see [4]) that any recursively enumerable language L is generated by a grammar G in the Geffert normal form, i.e., by a grammar of the form

$$G = (\{S, A, B, C\}, T, P \cup \{ABC \to \varepsilon\}, S)\,,$$

where P contains context-free productions of the form

$$\begin{aligned} S &\to uSa \\ S &\to uSv \\ S &\to uv \end{aligned}$$

where $u \in \{A, AB\}^*$, $v \in \{BC, C\}^*$, and $a \in T$.

In addition, any terminal derivation in G is of the form $S \Rightarrow^* w_1w_2w$ by productions from P, where $w_1 \in \{A, AB\}^*$, $w_2 \in \{BC, C\}^*$, $w \in T^*$, and $w_1w_2w \Rightarrow^* w$ is derived by $ABC \to \varepsilon$.

Clearly, G is a grammar system with only one component. Set the control language to be $\{1\}^*$. Then, the theorem holds. □

Study of Other Derivation Modes

The situation is analogical when regarding other derivation modes. Similar argumentation could be used for all modes $f \in \{\leq k, =k, \geq k \,:\, k \geq 1\}$. Therefore, the following theorem holds.

Theorem 5.9. $\mathbf{RE} = {}_{nonter}GS_f^{\mathbf{REG}}(1)$ *for all* $f \in \{t\} \cup \{\leq k, =k, \geq k \,:\, k \geq 1\}$.

5.3 Restriction on Number of Nonterminal Blocks of Limited Length

The following example shows that, in contrast with the prefix restriction, restriction studied in this section does not decrease the generative power of regulated grammar systems to the family of context-free languages.

Example 5.10. Consider a grammar system $\Gamma = (N, T, S, P_1, P_2, P_3, P_4)$, where

1. $N = \{S, A, A', B, B', C, C'\}$,
2. $T = \{a, b, c\}$,
3. $P_1 = \{S \to ABC\}$,
4. $P_2 = \{A \to aA', B \to bB', C \to cC'\}$,
5. $P_3 = \{A' \to A, B' \to B, C' \to C\}$, and
6. $P_4 = \{A \to \varepsilon, B \to \varepsilon, C \to \varepsilon\}$.

Observe that for $C = \{1\}\{2,3\}^*\{4\}$, ${}_{\text{nonter}}\mathcal{L}_t^C(\Gamma, 3, 3) = \{a^n b^n c^n : n \geq 0\}$, which is not context-free.

We show that the family of languages generated by grammar systems, when a derivation restriction studied in this chapter is applied, is a subset of the family of state languages. More precisely, we show that any grammar system whose sentential forms are regulated by constant m can be also generated by a state grammar of index m. This implies that such a state grammar is also of degree m. As a result, an infinite hierarchy of language families is established (follows from the theorem 2.36). The proof shows these facts by a chain of inclusions.

Firstly, we show that for any regulated grammar system which is limited by a constant h, we can construct a regulated grammar system which will generate the same language in the way that all sentential forms during derivation will contain nonterminal blocks of length one.

Next, as we are able to construct a grammar system which generates sentential forms where each nonterminal block is of length one, we can consider only derivations which are leftmost with respect to the set of productions currently used. This follows from the fact that components of a grammar system now contain only context-free productions and therefore, by analogy with context-free grammars, we can consider only left-most derivations without loss of generality.

Therefore, after showing these inclusions, we can prove the theorem by considering only grammar systems

- which generate sentential forms only with nonterminal blocks of length one and
- whose components perform only leftmost derivations

Lemma 5.11. *Let m, h be positive integers. Then,*

$$_{nonter}GS_t^{\mathbf{REG}}(m,h) = {}_{nonter}GS_t^{\mathbf{REG}}(m,1).$$

Idea. All strings in the sequence of derivation steps contain only a finite number of blocks of nonterminals and these blocks are also of a finite length. Hence, it is possible to represent each possible block by a single nonterminal and create an equivalent grammar system, which contains only context-free productions.

Lemma 5.12. *Let m be a positive integer. Then,*

$$_{nonter}GS_t^{\mathbf{REG}}(m,1) = {}^{left\text{-}most}_{nonter}GS_t^{\mathbf{REG}}(m,1).$$

Idea. A sequence of derivation steps in such a grammar system can be split into several subsequences, where each subsequence corresponds to a derivation using one component of the GS in the t-mode. The derivation steps are performed in the same manner as with an ordinary context-free grammar in each subsequence, so it is possible to take into account only the cases which consist exclusively of the left-most derivation.

Lemma 5.13. *Let m be positive integer. Then,* ${}^{left\text{-}most}_{nonter}GS_t^{\mathbf{REG}}(m,1) \subseteq \mathbf{SG}_m$.

Proof. Let $\alpha = x_0A_1x_1 \dots A_nx_n$, where $x_i \in T^*$, for all $0 \leq i \leq n$, $A_i \in N$, for all $1 \leq i \leq n$. Define $f(\alpha) = A_1 \dots A_n$.

Let $\Gamma = (N, T, S, P_1, P_2, \dots, P_n)$ be grammar system, where P_i contains only context-free productions for all $1 \leq i \leq n$, and $L_c = \mathcal{L}(M)$ be a control language, where $M = (Q, \Sigma, \delta, q_0, F)$ is a finite automaton. Introduce a state grammar $G_s = (N \cup T, W, T, P', s_0, S)$, where $W = \{[p, q, i, \alpha] : p \in \{\bigtriangledown, \bigtriangleup\}, q \in Q, i \in \Sigma \cup \{\varepsilon\}, \alpha \in N^*, 0 \leq |\alpha| \leq m\}$, and $s_0 = [\bigtriangleup, q_0, \varepsilon, S]$. Then, P' is constructed as follows:

1. for each $uAv \in N^+$, where $1 \leq |uAv| \leq m$, and $\mathrm{occur}(u, N_{left}(P_i)) = 0$, for each $A \to \beta \in P_i$, and $p \in Q$, add

$$[\bigtriangledown, p, i, uAv]A \to [\bigtriangleup, p, i, uf(\beta)v]\beta$$

$$[\bigtriangleup, p, i, uAv]A \to [\bigtriangleup, p, i, uf(\beta)v]\beta$$

   to P', if the following conditions are met:

   (a) $|uf(\beta)v| \leq m$

   (b) $uf(\beta)v \neq \varepsilon$ or $p \in F$

2. for each state $[\bigtriangleup, p, i, uAv]$ ($p \in Q$, $i \in \Sigma \cup \{\varepsilon\}$, $uAv \in V^+$ and $1 \leq |uAv| \leq m$) such that there is no rule generated by 1. with $[\bigtriangleup, p, i, uAv]$ on its left-hand side, for each $pi' \to q \in \delta$, add:

$$[\bigtriangleup, p, i, uAv]A \to [\bigtriangledown, q, i', uAv]A$$

   to P'

First, we prove that $\mathcal{L}(\Gamma) \subseteq \mathcal{L}(G_s)$ by induction on the number of productions used in the derivation of the sentential form in Γ.

Claim 5.14. *Let* $S \ {}^{1}_{m}\!\!\Rightarrow^{t} \ldots {}^{1}_{m}\!\!\Rightarrow^{t} {}^{1}_{m}\!\!\Rightarrow^{*}_{G_i} \omega\ [\tau i]$ *be a left-most derivation containing* k *productions in* Γ*, where* $\omega \in (N+T)^*$*,* $\tau i \in \Sigma^*$*,* $q_0\tau i \Rightarrow^* q$ *in* M*. Then,* $[\triangle, q_0, \varepsilon, S] \ {}_{m}\!\!\Rightarrow^{*} [\triangle, q, i, f(\omega)]\omega$ *in* G_s*.*

Proof. By induction on $k = 0, 1, \ldots$.

Basis: Let $k = 0$, so $S \ {}^{1}_{m}\!\!\Rightarrow^{0} S$ in Γ and $q_0 \Rightarrow^* q_0$ in M. Then, $[\triangle, q_0, \varepsilon, S]S \ {}_{m}\!\!\Rightarrow^{0} [\triangle, q_0, \varepsilon, S]S$ in G_s.

Induction hypothesis: Suppose that the claim holds for all $0 \leq m \leq k$, where k is a non-negative integer.

Induction step: Let $S \ {}^{1}_{m}\!\!\Rightarrow^{t} \ldots {}^{1}_{m}\!\!\Rightarrow^{t} {}^{1}_{m}\!\!\Rightarrow^{*}_{G_i} uAv \ {}^{1}_{m}\!\!\Rightarrow_{G_{i'}} u\beta v\ [\tau ij]$ (left-most) using $k+1$ productions in Γ, $q_0\tau ij \Rightarrow^* qj \Rightarrow^* r$ in M, and the last production applied during the derivation is $A \to \beta \in P_{i'}$, where $u, v \in (N \cup T)^*$, $\mathrm{occur}(u, N_{left}(P_{i'})) = 0$, and $\tau ij \in \Sigma^*$. By induction hypothesis,

$$[\triangle, q_0, \varepsilon, S]S \ {}_{m}\!\!\Rightarrow^{k} [\triangle, q, i, f(uAv)]uAv.$$

If $i \neq i'$, then, there exists

$$[\triangle, q, i, f(uAv)]A \to [\triangledown, r, i', f(uAv)]A$$

introduced in 2. Hence,

$$[\triangle, q, i, f(uAv)]uAv \Rightarrow [\triangledown, r, i', f(uAv)]uAv.$$

Otherwise, $r = q$. Since $\mathrm{occur}(u, N_{left}(P_{i'})) = 0$, we can use the production

$$[p, r, i', f(uAv)]A \to [\triangle, r, i', f(u\beta v)]\beta$$

introduced in 1. ($p \in \{\triangledown, \triangle\}$) to obtain

$$[p, r, i', f(uAv)]uAv \ {}_{m}\!\!\Rightarrow [\triangle, r, i', f(u\beta v)]u\beta v.$$

Therefore,

$$[\triangle, q_0, \varepsilon, S]S \ {}_{m}\!\!\Rightarrow^{k+1} [\triangle, r, i', f(u\beta v)]u\beta v,$$

which completes the induction step. □

Now, we prove that $\mathcal{L}(G_s) \subseteq \mathcal{L}(\Gamma)$ by induction on the number of productions constructed in 1. used in the derivation of the sentential form in G_s.

Claim 5.15. *Let* $[\triangle, q_0, \varepsilon, S]S \ {}_{m}\!\!\Rightarrow^{*} [\triangle, q, i, f(\omega)]\omega$ *in* G_s*, where* $\omega \in (N \cup T)^*$*. Then,* $S \ {}^{1}_{m}\!\!\Rightarrow^{t} \ldots {}^{1}_{m}\!\!\Rightarrow^{t} {}^{1}_{m}\!\!\Rightarrow^{*}_{G_i} \omega\ [\tau i]$ *using* k *productions in* Γ*, and* $q_0\tau i \Rightarrow^* q$ *in* M*, where* $k \geq 0$*.*

Proof. By induction on $k = 0, 1, \ldots$.

Basis: Let $k = 0$, so $[\triangle, q_0, \varepsilon, S]S \ {}_{m}\!\!\Rightarrow^{0} [\triangle, q_0, \varepsilon, S]S$ in G_s. Then $S \ {}^{1}_{m}\!\!\Rightarrow^{0} S$ in Γ, and $q_0 \Rightarrow^* q_0$ in M.

Induction hypothesis: Suppose that the claim holds for all derivation sequences containing no more than k productions.

Induction step: Consider a derivation

$$[\triangle, q_0, \varepsilon, S]S \ {}_m\Rightarrow^k [\triangle, q, i, f(uAv)]uAv \ {}_m\Rightarrow^* [\triangle, r, i', f(u\beta v)]u\beta v$$

in G_s, and the last production used during the derivation is

$$[\triangle, r, i', f(uAv)]A \to [\triangle, r, i', f(u\beta v)]\beta,$$

where $u, v \in (N + T)^*$. By the induction hypothesis,

$$S \ {}^1_m\!\Rightarrow^t \ldots \ {}^1_m\!\Rightarrow^t {}^1_m\!\Rightarrow^*_{G_i} uAv \ [\tau' i]$$

using k productions in Γ, and

$$q_0\tau' i \Rightarrow^* q$$

in M. There exists $A \to \beta \in P_{i'}$ by 1. of the construction. Moreover, $\mathrm{occur}(u, N_{left}(P_{i'})) = 0$. Then,

$$uAv \ {}^1_m\!\Rightarrow u\beta v$$

in Γ and

$$qj \Rightarrow^* r$$

in M, where $j \in \Sigma^*$. If the previous production was constructed in 1., then $i' = i$ and $j = \varepsilon$, else $j = i'$. In either case, $\tau' ij = \tau i'$, where $\tau \in \Sigma^*$. Therefore

$$S \ {}^1_m\!\Rightarrow^t \ldots \ {}^1_m\!\Rightarrow^t {}^1_m\!\Rightarrow^* u\beta v \ [\tau i']$$

using $k + 1$ productions in Γ and

$$q_0\tau i' \Rightarrow^* r$$

in M, and the induction holds for $k + 1$. □

Now consider, that the terminal string can be generated by G_s only in those cases, where M reaches a final state in Γ. Then, as a result of the previous claims, the lemma 5.13 holds. □

Theorem 5.16. *Let m, h be positive integers. Then, $\mathbf{SG}_m \supseteq {}_{nonter}GS_t^{\mathbf{REG}}(m, h)$.*

The theorem 5.16 is a direct consequence of the lemmata 5.11–5.13.

Study of Other Derivation Modes

Situation is analogical to prefix restriction regarding other derivation modes. Minor alterations of construction rules ensure correct simulation of restricted grammar system. Therefore, the following theorem holds.

Theorem 5.17. *Let m, h be positive integers. Then, $\mathbf{SG}_m \supseteq {}_{nonter}GS_f^{\mathbf{REG}}(m, h)$ for all $f \in \{t\} \cup \{\leq k, =k, \geq k \ : \ k \geq 1\}$.*

Chapter 6

Conclusion

Three types of derivation restrictions of grammar systems have been studied in this thesis. We dealt with grammar systems whose components were type-0 grammars. Moreover, grammar systems were allowed to be regulated by regular languages. Such grammar systems can generate all type-0 languages. However, we proved that only one of these restrictions does not change the generative power at all, while the other two restrictions do.

The first restriction studied, which restricts derivations to finite prefix of the first continuous block of nonterminals, decreases the generative power to the power of context-free languages. A formal proof shows this by a construction of a pushdown automata which simulates a given grammar system restricted in that way.

Second restriction prevents the generation of sentential forms which would contain more than a given number of blocks of nonterminals. This restrictions does not decrease the generative power. This is proved by showing that each phrase-structure grammar can be transformed to a grammar that will never generate a sentential form which has more than one continuous block of nonterminals.

Last restriction forbids a generation of sentential forms which contain more than m nonterminal blocks and where the length of each block is not greater than h. The proof shows that the generative power of grammar systems decreases when applying this kind of restriction, namely to the power which is no greater than the generative power of state languages. This is done by showing that we are able to construct a state grammar of index m for any grammar system restricted in that way—by constants m and h.

When considering practical issues, the third result is perhaps the most interesting. One of the main purposes for an introduction of new models and restrictions is to find a model which would make capturing of context properties of programming languages possible. Therefore, models whose generative power is between context-free and context-sensitive languages have been most desired. From this point of view, the third result is the most promising. The question what exactly the generative power of grammar systems with the third derivation restriction is remains an open problem.

Bibliography

[1] B. S. Baker. Context-sesitive grammars generating context-free languages. In M. Nivat, editor, *Automata, Languages and Programming*, pages 501–506. North-Holland, Amsterdam, 1972.

[2] R. V. Book. Terminal context in context-sensitive grammars. *SIAM Journal of Computing*, 1:20–30, 1972.

[3] G. Paun E. Csuhaj-Varju, J. Kelemen and J. Dassow, editors. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach Science Publishers, Inc., Newark, NJ, USA, 1994.

[4] V. Geffert. Context-free-like forms for the phrase-structure grammars. In M. Chytil, L. Janiga, and V. Koubek, editors, *Mathematical Foundations of Computer Science*, volume 324 of *Lecture Notes in Computer Science*, pages 309–317. Springer-Verlag, 1988.

[5] S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages*. Elsevier Science Inc., New York, NY, USA, 1975.

[6] S. Ginsburg and S. Greibach. Mappings which preserve context-sensitive languages. *Information and Control*, 9:563–582, 1966.

[7] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[8] T. Kasai. An hierarchy between context-free and context-sensitive languages. *J. Comput. Syst. Sci.*, 4(5):492–508, 1970.

[9] G. Matthews. A note on symmetry in phrase structure grammars. *Information and Control*, 7:360–365, 1964.

[10] G. Matthews. Two-way languages. *Information and Control*, 10:111–119, 1967.

[11] A. Meduna. *Automata and languages: theory and applications*. Springer-Verlag, London, UK, 2000.

[12] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 1. Springer-Verlag, Berlin, 1997.

[13] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 2. Springer-Verlag, Berlin, 1997.